In the Claims:

Please amend claims 2, 3, 4, 16, 19, 46-55, and 58 as indicated below.

1. (Original) A method of providing non-blocking multi-target transactions in a computer system, the method comprising:

defining plural transactionable locations, wherein individual ones of the transactionable locations encode respective values and are owned by no more than one transaction at any given point in a multithreaded computation;

for a particular multi-target transaction of the multithreaded computation, attempting to acquire ownership of each of the transactionable locations targeted thereby, wherein the ownership acquiring wrests ownership from another transaction, if any, that owns the targeted transactionable location; and

once ownership of each of the targeted transactionable locations has been acquired, attempting to commit the particular multi-target transaction using a single-target synchronization primitive to ensure that, at the commit, the particular multi-target transaction continues to own each of the targeted transactionable locations, wherein individual ones of the multi-target transactions do not contribute to progress of another.

2. (Currently amended) The method of claim 1,

wherein the ownership wresting employs a single-target synchronization primitive to change status of the wrested-from transaction to be incompatible with a commit thereof.

3. (Currently amended) The method of claim 2,

wherein, as a result of the status change, the wrested-from transaction fails and retries.

4. (Currently amended) The method of claim 2,
wherein the wrested-from transaction is itself a multi-target transaction.

5. (Original) The method of claim 1, further comprising:
on failure of the commit attempt, reacquiring ownership of each targeted transactionable location and retrying.

6. (Original) The method of claim 1,
wherein no transaction may prevent another from wresting therefrom ownership of transactionable locations targeted by the active transaction.

7. (Original) The method of claim 1,
wherein the ownership acquiring employs a single-target synchronization primitive to update the ownership of the targeted transactionable location.

8. (Original) The method of claim 1,
wherein each encoding of a transactionable location is atomically updateable using a single-target synchronization primitive.

9. (Original) The method of claim 1,
wherein the individual transactionable location encodings further include an identification of the owning transaction's corresponding value for the transactionable location.

10. (Original) The method of claim 1, further comprising:
accessing values corresponding to individual ones of the transactionable locations using a wait-free load operation.

11. (Original) The method of claim 1,
wherein the transactionable locations directly encode the respective values.

12. (Original) The method of claim 1,
wherein the transactionable locations are indirectly referenced.

13. (Original) The method of claim 1,
wherein the transactionable locations are encoded in storage managed using a nonblocking memory management technique.

14. (Original) The method of claim 1,
wherein the transactionable locations, if unowned, directly encode the respective values and otherwise encode a reference to the owning transaction.

15. (Original) The method of claim 1,
wherein the single-target synchronization primitive employs a Compare-And-Swap (CAS) operation.

16. (Currently amended) The method of claim 1,
wherein the single-target synchronization primitive employs a Load-Linked (LL) and Store-Conditional (SC) operation pair.

17. (Original) The method of claim 1,
wherein the single-target of the single-target synchronization primitive includes at least a value and a transaction identifier encoded integrally therewith.

18. (Original) The method of claim 1,
wherein the multi-target transaction has semantics of a multi-target compare and swap (NCAS) operation.

19. (Currently amended) The method of claim 1,

embodied in operation of an application programming interface (API) that includes a load operation and a[[n]] multi-target compare and swap (NCAS) operation.

20. (Original) The method of claim 19,
wherein the load operation is wait-free.

21. (Original) The method of claim 1,
embodied in operation of an application programming interface (API) that provides transactional memory.

22. (Original) An implementation of non-blocking, multi-target transactions that employs instances of one or more single-target synchronization primitives to acquire, for a particular multi-target transaction, ownership of targeted transactionable locations and to ensure that, at commit, the particular multi-target transaction continues to own each of the targeted transactionable locations, wherein individual ones of the multi-target transactions do not contribute to progress of others.

23. (Original) The implementation of claim 22, embodied as software encoded in one or more computer readable media and that, on execution as part of a concurrent computation, invokes the multi-target transactions.

24. (Original) The implementation of claim 22,
wherein the ownership acquiring, when performed by a first one of the multi-target transactions, wrests ownership from respective other ones of the multi-target transactions, if any, that own respective ones of the targeted transactionable locations.

25. (Original) The implementation of claim 24,

wherein the wresting employs an instance of a single-target synchronization primitive to change status of a wrested-from transaction to be incompatible with a commit thereof.

26. (Original) The implementation of claim 25,
wherein, as a result of the status change, the wrested-from transaction eventually fails and retries.

27. (Original) The implementation of claim 22,
wherein no transaction may prevent another from wresting therefrom ownership of transactionable locations targeted by the active transaction.

28. (Original) The implementation of claim 22,
wherein the transactionable locations directly encode the respective values.

29. (Original) The implementation of claim 22,
wherein the transactionable locations are indirectly referenced.

30. The implementation of claim 22,
wherein the transactionable locations are encoded in storage managed using a nonblocking memory management technique.

31. (Original) The implementation of claim 22,
wherein the transactionable locations, if unowned, directly encode the respective values and otherwise encode a reference to the owning transaction.

32. (Original) The implementation of claim 22,
wherein at least some instances of the single-target synchronization primitive employ a Compare-And-Swap (CAS) operation.

33. (Original) The implementation of claim 22,

wherein at least some instances of the single-target synchronization primitive employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

34. (Original) The implementation of claim 22,
wherein at least some of the multi-target transaction have semantics of a multi-target compare and swap (NCAS) operation.

35. (Original) The implementation of claim 22,
embodied as software that includes a functional encoding of operations concurrently executable by one or more processors to operate on state of the transactionable locations.

36. (Original) The implementation of claim 22,
wherein at least some of the multi-target transactions are defined by an application programming interface (API) that includes a load operation and a multi-target compare and swap (NCAS) operation.

37. (Original) The implementation of claim 22,
wherein at least some of the multi-target transactions are defined by an application programming interface (API) that provides transactional memory.

38. (Original) The implementation of claim 22,
wherein the multi-target transactions are obstruction-free, though not wait-free or lock-free.

39. (Original) The implementation of claim 22,
wherein the implementation does not itself guarantee that at least one interfering concurrently executed multi-target transactions makes progress.

40. (Original) The implementation of claim 22,

wherein a contention management facility is employed to facilitate progress in a concurrent computation.

41. (Original) The implementation of claim 40,

wherein operation of the contention management facility ensures progress of the concurrent computation.

42. (Original) The implementation of claim 40,

wherein the contention management facility is modular such that alternative contention management strategies may be employed without affecting correctness of the implementation.

43. (Original) The implementation of claim 40,

wherein the contention management facility allows changes in contention management strategy during a course of the concurrent computation.

44. (Original) A computer program product encoding, the encoding including:

instructions executable on a shared memory multiprocessor as a multithreaded computation that includes respective instances of a non-blocking, multi-target transaction that employs a single-target synchronization primitive to acquire, on execution of a particular multi-target transaction, ownership of targeted transactional locations in the shared memory and to ensure that, at commit thereof, the particular multi-target transaction continues to own each of the targeted transactionable locations,

wherein no execution of one of the multi-target transaction instances contributes to progress of another.

45. (Original) The computer program product encoding of claim 44,

wherein the ownership acquiring, when performed by execution of a first instance of the multi-target transaction, wrests ownership from respective other instances of the multi-target transaction, if any, that own respective ones of the targeted transactional locations.

46. (Currently amended) A computer readable storage medium encoding at least a portion of a nonblocking, multi-target transaction implementation, the encoding comprising:

a definition of a transactionable location instantiable in shared memory to individually encapsulate values that may be targeted by concurrent executions of the multi-target transaction; and

a functional encoding of the multi-target transaction that upon execution of a particular instance thereof, attempts to acquire ownership of each of the transactionable locations targeted thereby and, once ownership of each of the targeted transactionable locations has been acquired, attempts to commit the particular instance using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the targeted transactionable locations, wherein execution of no one of the multi-target transaction instances contributes to progress of another.

47. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein the ownership acquiring wrests ownership from another transaction, if any, that owns the targeted transactionable location.

48. (Currently amended) The ~~encoding~~ storage medium of claim 47,
wherein the another transaction is another concurrently executing instance of the multi-target transaction.

49. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Compare-And-Swap (CAS) operation.

50. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

51. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein the single-target of the single-target synchronization primitive includes a value and an owning transaction identifier encoded integrally therewith.

52. (Currently amended) The ~~encoding~~ storage medium of claim 46,
embodied as an application programming interface software component combinable with program code to facilitate execution of the program code as a multithreaded computation.

53. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein the multi-target transaction implements a multi-target compare and swap operation (NCAS).

54. (Currently amended) The ~~encoding~~ storage medium of claim 46,
wherein the multi-target transaction implements transactional memory.

55. (Currently amended) ~~A encoding~~ The storage medium of claim 46,
wherein the computer readable storage medium includes at least one medium selected from the set of a disk, a tape [[or]] and another magnetic, optical, or electronic storage medium ~~and a network, wireline, wireless or other communications medium~~.

56. (Original) An apparatus comprising:

one or more processors;

one or more data stores addressable by each of the one or more processors; and

means for coordinating concurrent non-blocking execution, by the one or more processors, of multi-target transactions that attempt to acquire ownership of each of transactionable locations targeted thereby and, once ownership of each of the targeted transactionable locations has been acquired, attempt to commit the particular instance using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the targeted transactionable locations, wherein none of the multi-target transaction contributes to progress of another.

57. (Original) The apparatus of claim 56, further comprising:

means for wresting ownership from another transaction, if any, that owns one of the targeted transactionable locations.

58. (Currently amended) The apparatus of claim 56,

wherein the wresting means includes means for ensuring that status of the wrested-from transaction is incompatible with a successful commit thereof.

59. (Original) The apparatus of claim 56, further comprising:

means for managing contention between interfering executions of the multi-target transactions.